**Assignment Number 07**

**Name:** Mihir Unmesh Patil

**Roll NO**: TYCOC213

**Batch:** C/C-3

**CODE:**

#include <stdio.h>

#include <stdbool.h>

bool is\_page\_present(*int* *frames*[], *int* *capacity*, *int* *page*) {

    for (*int* i = 0; i < *capacity*; i++) {

        if (*frames*[i] == *page*) return true;

    }

    return false;

}

*void* print\_frames(*int* *frames*[], *int* *capacity*, *int* *page*, bool *page\_fault*) {

    printf("Reference: %d | Frames: ", *page*);

    for (*int* i = 0; i < *capacity*; i++) {

        printf("%d ", *frames*[i]);

    }

    printf("\t");

    printf("| Page Fault: %s\n", *page\_fault* ? "Yes" : "No");

}

*int* run\_algorithm(*int* *reference\_string*[], *int* *n*, *int* *capacity*, *char*\* *name*,

*void* (\**replace\_page*)(*int* frames[], *int* capacity, *int* page, *int* reference\_string[], *int* n)) {

*int* frames[*capacity*];

    for (*int* i = 0; i < *capacity*; i++) {

        frames[i] = -1;

    }

    printf("%s Page Replacement:\n", *name*);

*int* page\_faults = 0;

    for (*int* i = 0; i < *n*; i++) {

        bool page\_fault = !is\_page\_present(frames, *capacity*, *reference\_string*[i]);

        if (page\_fault) {

            page\_faults++;

*replace\_page*(frames, *capacity*, *reference\_string*[i], *reference\_string*, *n*);

        }

        print\_frames(frames, *capacity*, *reference\_string*[i], page\_fault);

    }

    return page\_faults;

}

*void* replace\_page\_fcfs(*int* *frames*[], *int* *capacity*, *int* *page*, *int* *reference\_string*[], *int* *n*) {

    static *int* frame\_index = 0;

*frames*[frame\_index] = *page*;

    frame\_index = (frame\_index + 1) % *capacity*;

}

*void* replace\_page\_lru(*int* *frames*[], *int* *capacity*, *int* *page*, *int* *reference\_string*[], *int* *n*) {

    static *int* time[10];

    static *int* current\_time = 0;

    for (*int* i = 0; i < *capacity*; i++) {

        if (*frames*[i] == -1) {

*frames*[i] = *page*;

            time[i] = ++current\_time;

            return;

        }

    }

*int* lru\_index = 0;

    for (*int* i = 1; i < *capacity*; i++) {

        if (time[i] < time[lru\_index]) lru\_index = i;

    }

*frames*[lru\_index] = *page*;

    time[lru\_index] = ++current\_time;

}

*void* replace\_page\_optimal(*int* *frames*[], *int* *capacity*, *int* *page*, *int* *reference\_string*[], *int* *n*) {

    for (*int* i = 0; i < *capacity*; i++) {

        if (*frames*[i] == -1) {

*frames*[i] = *page*;

            return;

        }

    }

*int* farthest = -1;

*int* replace\_index = -1;

    for (*int* i = 0; i < *capacity*; i++) {

        bool found = false;

        for (*int* j = *n* + 1; j < *n*; j++) {

            if (*frames*[i] == *reference\_string*[j]) {

                if (j > farthest) {

                    farthest = j;

                    replace\_index = i;

                }

                found = true;

                break;

            }

        }

        if (!found) {

            replace\_index = i;

            break;

        }

    }

    if (replace\_index == -1) replace\_index = 0;

*frames*[replace\_index] = *page*;

}

*int* main() {

*int* reference\_string[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3};

    const size\_t n = sizeof(reference\_string) / sizeof(reference\_string[0]);

*int* capacity = 3;

*int* fcfs\_faults = run\_algorithm(reference\_string, n, capacity, "FCFS", replace\_page\_fcfs);

    printf("\nFCFS Total Page Faults: %d\n", fcfs\_faults);

*int* lru\_faults = run\_algorithm(reference\_string, n, capacity, "LRU", replace\_page\_lru);

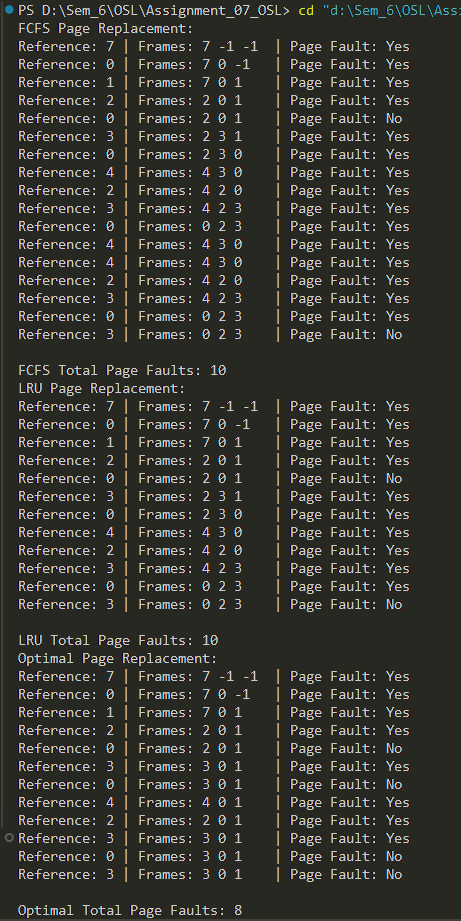
    printf("\nLRU Total Page Faults: %d\n", lru\_faults);

*int* optimal\_faults = run\_algorithm(reference\_string, n, capacity, "Optimal", replace\_page\_optimal);

    printf("\nOptimal Total Page Faults: %d\n", optimal\_faults);

    return 0;

}

OUTPUT: